Applicant respectfully traverses the rejections with arguments as set forth below. Applicant appreciates the indication of allowability as to numerous dependent claims. Applicant has amended claims 1-3, 6, 22, 24, 27, 51, 52, 59, 74, 90, 94, 125 as shown in the Appendix and discussed below, not for the purpose of patentably distinguishing over the cited references, but to improve the clarity of distinctions that applicant believes are already present in the claims.

Applicant and his patent counsel remain willing and available to interview with the Examiner to discuss the prior art rejections, the relevant claim language, and applicant's arguments traversing the rejections. Applicant can also arrange for a demonstration of his system to the examiner.

In sum, the Truong patent is no more relevant to the present invention than the previously cited Massena patent. While Truong does disclose a web-based editor, it is a text editor only and does not permit editing of fully functional, running web applications. Truong does not even provide WYSIWYG editing, where the page being edited looks similar to the running application without any functionality. For at least the foregoing reasons, and in light of the arguments below, applicant submits that the examiner is in error regarding the applicability of the cited patents to applicant's claims, and reconsideration is requested.

II. DISCUSSION OF THE CITED ART

The examiner rejected the claims based primarily the Truong patent. Applicant acknowledges that Truong allows for editing of text files stored on web servers by using a web browser. However, Truong's editor does not even edit functional web application in WYSIWYG mode, but shows the source code of the pages during editing. As is well known, the source code of an HTML page looks dramatically different than the actual page being displayed by the browser, and scripts and other dynamic features are not executed when the source code is displayed. In contrast, the claimed invention displays pages for editing which look similar to the pages displayed to the end user, and the pages remain functional during editing. This important distinction is specifically reflected in the claim language used by applicant, as discussed in greater detail below.

For some of the claims, the examiner also cited the Prithvirag patent, proposing to combine the feature of document templates as disclosed in Pritvirag to Truong's editor and

components. However, since Truong does not disclose an editor or components as claimed by applicant, the proposed combination is ineffective to teach or suggest any of applicant's claims.

A. Editing of Functional Applications

Truong's editor shows and edits files in text mode. It does not show HTML pages in a WYSIWYG mode nor does it execute an application being edited during editing. In fact Trunong's editor handles any kind of text files and is not specifically constructed for HTML pages. This is explained in Truong's abstract, wherein Truong specifically talks about editing of files stored on a remote internet server. Truong does not restrict the kind of files that can be edited. Without being specific information on how to render a file, an editor can not display the file in a WYSIWYG manner nor keep the file functional. Figure 5 of Truong makes this explicitly clear. Truong labeled Figure 5 in column 4 line 10-14 as "*an exemplary output display of the remote editor system ... illustrating the display of a file for editing.*" Figure 5 shows an internet page that is part of the Truong's editor displayed in a browser. This page contains a text area containing the source code of a CGI script. This illustrates that the editor is not made to specifically handle HTML pages since it is used to edit a script in this example, that the editor displays a source code text view and not a WYSIWYG view of the file being edited, and that the editor shows the source code of a script rather than executing the script and showing the output. Truong column 10 line 45 to 50 also provides support for this view stating "*providing the text of the selected file to the web browser for editing as shown in figure 5 ... the text may be edited at the web browser.*"

In addition, applicant's editor shows an application being developed in a functional way, i.e., it executes the application including server side and client side scripts during editing and shows the output of this processing to the developer, just as it is shown to an end user (except for the addition of editing features). In contrast, Truong's editor shows the source text without execution. This distinction is specifically reflected in the claim language used by applicant, for example, in claim 1: "*functional application pages with editing features*" and "*allowing the user to work on a functional application during development;*" in claim 26: "*appears and functions similar to the first document;*" in claim 59: "*the editor*"..."*displaying at least some information items contained on said generated document;*" in claim 90: "*scripts contained in*

said document remain functional;" and in claim 125: "*a method for editing an application ... comprising the steps of running the application.*"

B. Components

Applicant's editor has specific functions for editing components. In contrast, Truong's editor works on text, and does not have specific functions to operate on components.

The examiner cited Truong column 2 lines 1-5 as disclosing components. The cited portion discloses conventional client side HTML elements used by the internal implementation of the editor. These HTML elements work completely on the client side, not on server side as applpicant's ISSCs. This distinction is expressed using claim language as "*the server computer comprising a plurality of components*" as recited in claim 6, and "*a plurality of components for execution on the server*" as recited in claim 114.

In addition, Truong's editor does not function to operate on these HTML elements nor on any kind of components. Truong column 10 lines 47 to 50 states "*the text may be edited at the web browser using editing features*" which makes clear the Truong's editor works on text. So does Figure 5, showing a text area input box of a browser. In addition, Figure 5 does not show any user interface for editing components. This distinction is expressed using claim language such as "*an editor ... for inserting, deleting, and modifying components*" as recited in claim 22, "*a user interface for editing functions used for maintaing components on document templates*" as recited in claim 51, and "*an editor capable of performing edit functions maintaing components on document templates*" as recited in claim 74.

III. THE PENDING CLAIMS ARE PATENTABLE

The examiner rejected claims 1, 22, 90-96, 114-119 and 121-127 as unpatentable over Truong, and rejected claims 2-8, 23-26, 30-33, 41-43, 51-55, 59-72, 74-77, 82, 84-86 and 88 as unpatentable over the combination of Truong and Prithvirag. However, applicant respectfully traverses these rejections for the reasons discussed below. Further, the Examiner indicated that claims 27-29, 56-58, 73, 78-81, 83, 87, 89 and 120 would be allowable if rewritten to incorporate the limitations of respective base claims. However, for the reasons stated herein, applicant believes the base claims to be patentable, therefore applicant requests that the rejection be withdrawn.

A. CLAIMS 1, 22, 90-96, 114-119, 121-127

Claim 1

Claim 1 is independent claim directed to a "*software development system*" for an online data network, wherein the user (client) computer runs a browser program to interpret web pages posted by the server. The system includes "*a page generator*" and "*an editor*," whereby the page generator runs the application being developed during editing. This allows the developer to see application pages during editing that look and function similar to the end users' view of the application. In contrast, Truong's editor does not follow the WYSIWYG principle, but instead shows the source code of an application during editing. This can be seen from Truong's Figure 5 as discussed in section II above in more detail.

This distinction is specifically reflected in the language of claim 1, as set forth below. Applicant has amended claim 1 to clarify that the pages displayed by the browser are identical to the pages produced by the page generator by running the application being developed, and to clarify the use of the editing features.

> 1. A software development system for applications that run on a data network which couples a server computer and a client computer, wherein the client computer runs a browser program, comprising
>
> a page generator running an application being developed and sending generated documents to the browser for display as pages including additional editing features for interpretation by the browser program;
>
> an editor directly operating on the pages displayed by the browser via the editing features, thereby allowing the user to work on a functional application during development.

In contrast, Truong's page generator does not generate pages of the application being developed, but pages of the editor only. Applicant has amended claim 1 for improved clarity with regard to the recitation of "functional application pages" by changing to "*running an application being developed and sending generated documents to the browser for display as pages including additional editing features for interpretation by the browser program.*" As shown in Truong's Figure 5, the editor shows the source code of the application developed, i.e.,

the application developed is not functional. In contrast, applicant's editor executes scripts during editing and thereby makes the developed application appear functional during editing.

The examiner cited Truong column 1 line 65-67 and column 2 line 1-10 and line 17-30 as disclosing a document generator capable of generating functional application pages. Applicant respectfully disagrees. Truong discloses information about well known tools such as browsers, web pages, URLs, forms and page generators, but does not disclose any specific properties of a page generator, especially not a page generator that generates functional application pages during editing. Applicant therefore submits that Truong does not disclose a page generator with the claimed properties.

The claim also requires allowing the user to work on a functional application during development, which requires execution of the application being developed during editing. In contrast, Truong column 10 lines 45-50 and Figure 5 reveal that the text of the selected file is being edited without execution.

For these reasons, applicant believes that claim 1 is patentable over the cited art and requests that the Examiner withdraw the rejection and allow the claim.

Claim 22

Claim 22 requires that the editor be capable of inserting, deleting and modifying of components. Truong does not disclose components as that term is used by applicant, nor does Truong disclose editing of components, as discussed in Section II above. The examiner cited Truong column 10 lines 45-50 as disclosing editing features such as delete, select, search, copy, paste, and the like; however the same portion of Troung makes clear that these are functions for editing text, not components. In addition, claim 22 requires that the editor and the page generator work on document templates. In his reasoning with respect to claim 6, the examiner admits that "Truong does not explicitly disclose a plurality of document templates." For these reasons, applicant believes that claim 22 is patentable over the cited art and requests that the Examiner withdraw the rejection and allow the claim. Applicant amended claim 22 to make clear that components are executed during document generation.

Claim 90

The examiner cited Truong column 1 lines 65-67, column 2 lines 1-10 and 17-30, as disclosing an editor that allows the user to edit a document being displayed by the browser. The cited portion does not reveal details about Truong's editor, but instead provides background regarding web technology, such as form capable browsers. Such browsers can edit text contained inside form fields displayed on an HTML page. However, this is different from editing the HTML page itself. In fact, Truong column 10 lines 45-50 and Figure 5 reveal that Truong's editor is made for editing a file displayed inside a form field and not for editing the displayed HTML page itself.

The examiner also cited Truong column 7 lines 1-5 as disclosing scripts staying functional during editing. However, the cited portion reveals only that the browser is capable of processing scripts. This does not imply or teach or suggest that the file being edited is sent to the browser in such a way that the browser recognizes and executes the scripts during editing. In fact, Truong column 10 lines 45-50 discloses that the text of the file is sent to the browser in such a way that it can be edited as text, which implicitly means that scripts are not executed by the browser but are shown in source code form. In addition, Figure 5 of Truong shows editing of a script, which further supports applicant's contention that according to Truong, scripts in the selected file are shown as text and are not executed.

Applicant has amended claim 90 for improved clarity as shown below to stress that scripts remain functional during editing.

> 90. An editor for use with a web browser, the editor allowing the user to edit a document displayed by the browser, wherein clicking on said document displayed in the browser initiates editing functions, and scripts contained in said document remain functional during editing, the editor including a first software program for execution within the browser and for processing the clicking on said document.

Claims 91-96

Claims 91-96 all depend from claim 90. Thus, for all the reasons given with regard to claim 90, applicant submits that Claims 91-96 are likewise not taught or suggested by Truong.

With regard to claim 91, the examiner cited Truong column 6 lines 55 to 65 as disclosing the editor using a second browser window for showing information on elements contained in a

document. However, the cited portion discloses HTML elements, but does not talk about browser windows. In addition, the cited portion does not provide any details on the user interface of Truong's editor, merely information on the web browser used.

With regard to Claim 92, the examiner cited Truong column 10 lines 45 to 50 as disclosing modifying documents in cooperation with the first software program. However, the cited portion reveals transmitting the content of a file to the web browser. This is clearly different from modifying a document in cooperation with the first software program.

With regard to Claim 93, the examiner cited column 7 lines 20-30 as disclosing that a generated document looks similar to the original. The cited portion talks about generating a document for display to the user, but does not indicate that the generated document looks similar to any other document. In contrast, Truong's Figure 5 shows clearly that the editor shows the source code text, which looks, in the case of HTML documents, clearly different than the original view of an HTML document.

With regard to Claim 94, the examiner cited Truong column 2 lines 1-10 as disclosing instructions for generating browser code for components. The cited portion discusses various HTML elements, but does not disclose any generation process. Applicant amended claim 94 in order to clarify that the third software program cooperates with the components generated by the browser code.

With regard to Claim 96, the examiner cited Truong column 8 lines 39 – 45 as disclosing that a link contained in a document stays functional. However, the cited portion reveals generation of a file selection HTML form. There is no written description as to how the file being edited is displayed. However, Figure 5 does shows that the file being edited is displayed as source text. This, however, implies that links contained in the source text are not functional as displayed.

Claim 114

Claim 114 is an independent claim. The examiner cited Truong column 10 lines 45-50 as disclosing a plurality of components for execution on the server wherein at least one of the components includes first features to cooperate with an editor in editing said components and second program instructions to generate browser code. However, applicant respectfully

disagrees. The cited portion of Truong teaches sending the text of a selected file to the browser and then using the built in editing functions of the browser to edit the text.The cited portion does not seem to disclose components of any kind, nor are there any of the claimed properties of the components, such as having features to cooperate with the editor, being editable, or containing specific program instructions.

The examiner also cited column 7 line 65 to column 8 line 15 as disclosing third program instructions on the server for generating generated documents, thereby calling second program instructions of components. However, the cited portion discloses generating an HTML page, then sending it to the browser and executing it. Applicant acknowledges that this discloses that the remote editor program generates HTML code, which implies that the remote editor program contains instructions for generating HTML code . In contrast, applicant's claim requires that instructions for generating HTML code be contained inside components that can be edited with the editor, which is clearly different from being contained in the editor itself.

Claims 115-119

Claims 115-119 are dependent from Claim 114. Thus, for all the reasons given with regard to claim 114, applicant submits that Claims 115-119 are likewise not taught or suggested by Truong.

With regard to claim 115, the examiner cited Truong column 10 lines 55-59 as disclosing that first features include instructions for passing information to the editor. According to claim 114, components for execution on the server include first features for cooperation with the editor. The cited portion reveals instructions for sending a file back to the server. These instructions are a part of the editor for execution on the client computer. In constrast, claim 115 requires instructions to be part of the components for execution on the server computer.

With regard to claim 116, the examiner cited Truong column 7 lines 60-67 as disclosing that part of the information is collected during execution of the components on the server. However, the cited portion describes execution of the editor, not of the components. Truong does not disclose any execution of components on the server. The examiner has apparently interpreted HTML text boxes and buttons (column 2 lines 1-5) as "components" in his discussion of claim 2. These HTML elements are part of the browser, however, and are not executed on the

server. In contrast, claim 116 requires components to be executed on the server thereby collecting information for the editor.

With regard to claim 118, the examiner cited Truong's Figure 5 as disclosing that the information includes attributes of the component. However, Figure 5 of Truong shows the user interface for the editor. The figure does not disclose any details about the information and therefore appears irrelevant.

With regard to claim 119, the examiner cited column 10 lines 45 to 50 as disclosing fifth features including fifth instructions that display additional editing features. The cited portions seems to reveal editing features as part of the editor and browser. In contrast the claim requires first features to include fifth instructions to display the editing features, whereby first features need to be included in the component according to claim 114, not inside the editor and not inside the browser.

Claims 121-127

Claims 121-127 are dependent from Claim 114. Thus, for all the reasons given with regard to claim 114, applicant submits that Claims 121-127 are likewise not taught or suggested by Truong.

With regard to claim 121, the examiner cited column 10 lines 45 to 50 as disclosing fifth features including extensions for use by the editor. The cited portions reveals various features as part of the editor and browser. In contrast, claim 121 requires the first features to include the extensions, whereby the first features are included in the components according to claim 114, not inside the editor and not inside the browser.

With regard to claim 122, the examiner cited Truong column 8 lines 65 to 67 as disclosing a web page for editing component attributes. The cited portion discusses the generation of web pages in general and does not disclose the generation of a web page for a specific purpose. In contrast, claim 122 specifically requires a page for editing component attributes.

With regard to claim 123, the examiner cited Truong column 6 lines 57-63 as disclosing components denoted on document templates using tag syntax. However, the cited portion discloses **HTML tags** denoted on document templates using tag syntax. **HTML tags** are

different from **components** as used by applicant in claim 114, since components are required to run on the server and generate browser code. Applicant submits that the cited portion does not disclose components that run on the server denoted on document templates using tag syntax.

With regard to claim 124, the examiner cited Truong column 8 lines 65 to 67 as disclosing components including instructions to generate browser code. The cited portion discloses that the remote editor program contains instructions to generate browser code. In contrast Claim 124 requires second program instructions to generate browser code and the base claim 114 requires that the second program be part of the components.

Claim 125

Claim 125 is an independent claim directed to a method for editing an application. The examiner cited Truong column 9 lines 1-5 as disclosing a method for editing an application comprising a step of running the application. In applicant's reading, the cited portion describes the first steps 100, 102 and 104 of a method of using the remote editing system. Step 100 is the start, step 102 involves executing a web browser, and step 104 involves requesting the network editor web page from the remote internet server. Only step 102 involves running a program, but it makes clear that the web browser is being run. In contrast, claim 125 requires that the application being edited and the application being run are identical. The browser resides in binary files at the client and is not stored in text files on the server, and therefore can not be edited with Truong's editor. Applicant therefore submits that Truong does not include a step of executing the application being edited during editing.

The examiner also cited Truong column 8 lines 65 to 67 as displaying a view of the generated document. In applicant's reading of the cited portion, Truong discloses displaying a generated document. The cited portion does not disclose that the generated document is a view of the document that was generated as the first step of the method by executing the application being edited.

The examiner cited Truong column 8 lines 36 to 38 as disclosing selecting a component of the source code of the application by clicking on selected portions of the view. However, the cited portion of Truong discloses selecting a file on a file selection form. The file selection form is part of the editor and not part of the application being edited. In contrast, claim 125 requires

the user to click on a selected portion of the view, whereby the view is a view of a document that was generated by the application.

The inventive editor is capable of executing the application while it is being edited. In contrast, Truong's editor shows the source code in text form only. This distinction is clearly represented in the claim language: "*a method for editing an application ... comprising steps of ... running the application.*"

Applicant has amended the claim to clarify that components are executed while the application is run. This rules out HTML tags as components.

Claims 126-127

Claims 126-127 are dependent from Claim 125, and for all the same reasons, applicant submits that Claims 126-127 are likewise not taught or suggested by Truong.

With regard to claim 126, the examiner cited Truong column 10 lines 45 to 50 as disclosing repeating the running and the displaying steps after applying a modification function. However, applicant submits that the cited portion does not refer to a step of running the application. With respect to claim 125, the examiner referred to steps 102 and 104 of Figure 3C as the running step. As clearly depicted in Figure 3C, these steps are not repeated. In contrast, claim 126 requires that the running step to be repeated.

B. CLAIMS 2-8, 23-26, 30-33, 41-43, 51-55, 59-72, 74-77, 82, 84-86, 88

Claims 2-5

Claims 2-5 are dependent from Claim 1, and for all the same reasons, applicant submits that claim 2-5 are not taught or suggested by Truong, either alone or in combination with Prithvirag.

With regard to claim 2, the examiner cited Truong column 2 line 1 to 5 as disclosing components and Truong column 10 line 47-50 as disclosing features to insert, modify and delete components. However, column 10 lines 47 to 50 discloses features to insert, modify and delete **characters** in the source text, not components. Claim 2 requires features to insert, modify and delete **components.** Applicant submits that inserting, deleting, and modifying **components**

involves different and more complex operations than inserting, deleting, and modifying of **characters.** Because the examiner cited HTML tags as components, applicant has amended the claim to clarify that components are executed by the page generator.

With regard to claim 3, the examiner cited Truong column 2 lines 1-5 as disclosing that at least one of the components reacts interactively on user input by executing instructions on the server. However, the cited portion lists HTML elements that are capable of **sending** information to the server. In contrast, claim 3 requires that the component themselves react and execute instructions on the server. In Truong, the HTML elements are interpreted by the browser and are not actually present on the server, which also means they cannot execute instructions on the server.

With regard to claim 4, the examiner cited applets as being nested components. However, claim 2 was amended to require that components be executed by the page generator. This is not the case for applets, since applets are specifically designed for execution by the browser. Therefore, in the context of claim 4, components cannot be interpreted as applets.

Claim 6

Claim 6 is an independent claim directed to a software development system that allows the user to create client server based applications by plugging together components. Applicant's invention provides components that can interactively communicate with the user on the client computer and that can execute instructions on the server as well.

The examiner cited Truong column 2 lines 1 to 5 as disclosing a plurality of components residing in the data store on the server including components that react interactively on user input by executing instructions on the server. However, Truong refers to browser built in HTML forms and HTML fields. Since these components are built into the browser, they reside in the data store of the client and they are executed on the client, not the server. HTML forms and fields can send data to the server, but they do not contain instructions for execution on the server. This should be clear since the complete browser with everything in it runs on the client only. In contrast, the claim language explicitly requires that the data store and the components reside on the server by saying "*the server computer further comprising a data store and a plurality of components residing in the data store.*" The claim language also requires that components

contain instructions for execution on the server by saying "*components that react interactively by executing instructions on the server.*"

The examiner also cites Truong column 5 lines 60 to 65. In applicants reading, this cited portion explains how internet servers send URLs and pages to each other and finally to a client. The cited portion does not disclose components.

The examiner also cites Truong Figure 3c item 128 as disclosing a data store. The cited portion states "*store each string as a variable,*" which indicates that data is stored on the server, but not the fact that the data store contains components.

Applicant amended claim 6 in order to clarify that the instructions a component reacts with are part of the component.

Claims 7-8

Claims 7 and 8 are dependent from Claim 6, and for all the same reasons, applicant submits that claims 7-8 are likewise not taught or suggested by Truong.

With regard to claim 7, the examiner refers to his reasoning to reject claim 5. However, claim 5 deals with a varying set of components. In contrast, claim 7 requires "*instructions for interactively editing selected components.*" As pointed out with respect to claim 2, Truong's editor does not have a specific function for "*interactively editing selected components*" - it just has a function for editing characters of the source code.

With regard to claim 8, the examiner cited applets as being nested components. However, independent claim 6 requires components to reside on the server and to execute instructions on the server. This is not the case for applets, since applets are specifically designed for execution on the client computer. Therefore, in the context of claim 6, components cannot be interpreted as applets

Claims 23-25

Claims 23-25 are dependent from claim 22, and for all the same reasons, applicant submits that Claim 23 is likewise not taught or suggested by Truong, either alone or in combination with Prithvirag.

With regard to claim 23, the examiner cited Truong column 2 lines 1 to 5 and column 10 lines 45-50 as disclosing that the editor operates functional applications in an edit mode thereby permitting editing directly in the web browser. However, the cited portion actually discloses editing of source text in the web browser, but not the operation of functional applications in an edit mode. In fact, Truong column 10 lines 45-50 states that the editor edits the source text and Figure 5 shows an example. In contrast, applicant's editor operates a functional application in an edit mode thereby permitting editing of generated pages directly in the browser.

With regard to claim 24, the examiner cited Truong column 10 line 45 to 50 as disclosing a component that can react on subsequent document requests by executing selected instructions. The cited portion actually reveals instructions of the editor. In contrast, the claim requires that the instructions belong to the component and be executed upon subsequent document requests.

Applicant has amended the claim to clarify that selected instructions are part of the component and that the request originated from the browser.

Claim 26

Claim 26 is an independent claim directed to a method for editing a document on a server by transforming it into a second document, by inserting handles or scripts, so that displaying the second document in an appropriate browsing program allows the user to edit the first document. The examiner rejected this claim by referring to his reasoning in the rejection of claim 6. However, claim 26 is different from claim 6 because it is specifically concerned with editing technology, and applicant submits that the reasoning for claim 6 is wholly inapplicable to claim 26.

As discussed above, Truong does not follow the WYIWYG principle but just shows the source code of an application being edited. Showing the source code also implies that scripts are not executed during editing, and so pages shown during editing do not function. In contrast, the claimed invention requires that the page during editing looks similar to the end user's view, and that pages stay functional during editing. This distinction is expressed using the claim language "*such that at least part of the second document appears and functions similar to the first document.*"

Claim 27:

Claim 27 dependent from claim 26 and has been indicated as allowable by the examiner. Applicant has amended the claim in to clarify that components are contained in the first document and are executed by the first software program, thereby avoiding confusion with HTML tags.

Claims 30-33

Claims 30-33 are dependent from Claim 26, and for all the same reasons, applicant submits that claims 30-33 are likewise not taught or suggested by Truong, either alone or in combination with Prithvirag.

With regard to claim 32, the examiner cited Truong column 9 lines 10-15 as disclosing features which permit editing of the first document wherein the features incorporate information regarding the first document into the second document. However, the cited portion actually discloses an editor input form which is being added to the first document. Although the input form does permit editing, the input form itself seems to be the same for each document being edited. So the form does not seem to be a feature that incorporates information regarding the first document. In contrast, claim 32 requires features that permit editing and that incorporate information regarding the first document into the second document.

With regard to claim 33, the examiner cites Truong at column 7 lines 10-30 as disclosing the sending of change requests for the first document to the server. However, the cited portion discloses generation and sending of information from the server computer to the client. In contrast, claim 33 describes sending information to the server. Truong column 10 lines 57 to 58 discloses sending information to the server. Truong sends the complete file being edited to the server. In contrast, it is claimed that change requests are sent to the server.

Claim 41-43

Claims 41-43 are indicated as rejected, however, the Examiner did not provide any specific reasoning. Applicant believes the claims are patentable for all the reasons discussed herein.

Claim 51

Claim 51 is an independent claim describing a system for editing components on web document templates. The examiner rejected this claim based on his reasoning with regard to claim 6. Applicant also refers to its discussion of claim 6 above. The main distinction between Truong's editor and the claimed invention is that Truong does not have components and his editor works on source text only. This distinction is represented in the claim language, for example, "*the system comprising*" and "*a user interface for editing functions used for maintaining components on document templates.*" The cited references do not disclose a user interface for editing functions for maintaining components. The editing functions disclosed in Truong at column 10 lines 47 to 50 are clearly identified as working on text "*the text may be edited at the web browser using editing features.*" In addition, Truong's Figure 5 shows the user interface of the editor but does not show a user interface for editing functions for maintaining components". Applicant has amended this claim to clarify that components encapsulate browser code.

Claims 52-55

Claims 52-55 are dependent from Claim 51, and for all the same reasons, applicant submits that claim 52 is likewise not taught or suggested by Truong, either alone or incombination with Prithvirag.

With regard to claim 52, the examiner cited Truong column 7 line 1 to 15 as disclosing components including forth program instructions including steps to generate browser code prior to transmission to the browser program. The cited portion discloses web browsers that can execute scripts. Scripts can indeed be used to generate browser code. The examiner cited Truong column 2 line 1 to 5 as disclosing components in the discussion of claim 2 and claim 6, identifying HTML textboxes and buttons as components. HTML text boxes and buttons however do not contain scripts for generation of browser code. In addition HTML element and script are clearly part of the browser. In contrast the claim explicitely requires the generation of browser code prior to the transmission to the first software program. Applicant amended claim 52 by substituting the language "for" instead of "prior to" in order to clarify the claim.

Claim 55 discloses edit functions working on components. As reasoned with respect to claim 2, Truong's edit functions work on **characters**, not on **components**.

Claim 59

Claim 59 is an independent claim directed to a software development system for dynamic web documents capable of displaying functional applications during editing. The examiner rejected this claim based on his reasoning towards claim 6. Applicant submits that claim 59 is quite different than claim 6 because it is specifically concerned with editing technology, and the reasoning for claim 6 is wholly inapplicable to claim 59.

As explained above, Truong's editor works on the source code of an application and consequently does not follow the WYSIWYG principle and does not execute the application during editing. In contrast, the inventive editor is capable of executing the application being developed during editing and so the application appears functional during editing.

This distinction is expressed by the following claim language, "*the editor program comprising first instructions for requesting the document generator to process a dynamic web document leading to a generated document.*" This language makes clear that the editor indeed executes the application during editing. Additional claim language, namely "*instructions to modify* ***the dynamic web document***" make clear that the dynamic web document that is being edited is the one being requested. Applicant amended the claim to clarify that the generated document looks and functions similar to the display document that is generated from the dynamic web document without edit mode. Applicant has amended claim 59 to clarify that the dynamic web document looks and functions similar to the generated display document.

Claims 60-72

Claims 60-72 are dependent from Claim 59, and for all the same reasons, applicant submits that Claims 60-72 are likewise not tought or suggested by Truong.

With regard to claim 61, the examiner cited Truong column 8 lines 39 to 50 as disclosing further instructions for execution during document generation to collect edit information for use by the editor. The cited portion describes document generation as part of the editor itself. According to claim 59, "the document generator" means the document generation usually

required for display of a dynamic web document. Claim 59 has been amended to make this fact very clear. In contrast, the cited portion is concerned with document generation taking place inside the editor.

With regard to claim 67, the examiner cited Truong column 8 lines 39 to 50 as disclosing that the view, except for editing features, looks similar to the end-user view of the generated document. However, the cited portion discloses the file selection form but does not give information on similarity. On the other hand, Figure 5 clearly indicates that Truong's editor shows the source code of a document. The source code of an HTML document does not look similar to its end-user view.

With regard to claim 70, the examiner referred to Claim 64 for reasoning. Claim 70 requires "position information on the components contained in the document template" while claim 64 does not. Applicant submits that Truong does not teach or disclose position information and therefore is not prior art for claim 70.

With regard to claim 72, the examiner cited Truong column 7 lines 59 to 67 as discloing initiating a reload in the browser. In applicant's reading, the cited portion discloses loading of the remote editor program, however, it does not seem to disclose initiating a reload in the browser. A reload in the browser means that the browser will load the same page a **second time**. In contrast, the cited portion discloases a **first time** load.

Claim 74

Claim 74 is an independent claim describing a software development system using components on web document templates. The examiner rejected this claim using his reasoning towards claim 6. Applicant refers to his discussing of claim 6 above. Truong's editor works on text only while applicant's editor contains specific editing functions for handling components. This distinction is represented in the claim language, namely "*an editor capable of performing edit functions maintaining components on document templates.*" None of the cited references disclose that Truong's editor is capable of performing edit functions maintaing components. The editing functions disclosed in Truong column 10 lines 47 to 50 clearly identify themselves as working on text "*the text may be edited at the web browser using editing features*".

Claims 75-77, 82, 84-86, 88

Claims 75-77, 82, 84-86 and 88 are dependent from claim 74, and for all the same reasons, applicant submits that claims 75-77, 82, 84-86 and 88 are likewise not tought or suggested by Truong.

With regard to claims 75 and 77, the examiner indicated them as rejected, however, the Examiner did not provide any specific reasoning supporting the rejection. Applicant believes the claims are patentable for all the reasons discussed herein.

With regard to claim 76, the examiner cited Truong column 6 lines 57 to 63 as disclosing that tag syntax is used to denote components on document templates. The cited portion reveals only that HTML tags are denoted using tag syntax on document templates. However, HTML tags are not components that generate browser code for transmission to the first software program as required by claim 74, nor are HTML tags components that cooperate with the first software program as required by claim 74. With regard to claim claim 82, the examiner did not provide reasons to support his rejection. However, Truong's Figure 5 clearly indicates that the editor shows the source code of an application being edited. In contrast, the claim requires the editor being able to take the varying set of components into account, which implies that the editor must be capable of showing various views of the same document.

With regard to claim 84, the examiner referred to claim 4 for his reasoning. However, claim 4 is concerned with nested components. In contrast, claim 84 is concerned with the set of components on the document template and on the generated document. Since claim 5 is concerned with a the set of components, applicant submits that for all the reasons given with claim 5, Claim 84 is likewise not tought or suggested by Truong.

The examiner's reasoning for claims 4 and 5 is based on the reasoning of the base claim 2. Here, the examiner interprets components as HTML tags. Claim 74, which is the base claim for claim 84, however, explicitly states that components include instructions to generate browser code prior to transmission to the first software program. This is clearly not the case for HTML tags. This means that the reasoning of claim 4 and claim 5 is not applicable to claim 84.

With regard to claim 86, the examiner cited Truong column 8 lines 20 to 35 as disclosing instructions to qualify names generated into the browser code with the unique identifier. The cited portion seems to disclose including file names into a generated page, however, the file

names are known before the generation process. In contrast, claim 86 requires a unique identifier be assigned to each component instance and this unique identifier be used to qualify the names.

C. CLAIMS 27-29, 56-58, 73, 78-81, 83, 87, 89 and 120

The Examiner indicated that claims 27-29, 56-58, 73, 78-81, 83, 87, 89 and 120 would be allowable if rewritten to incorporate the limitations of respective base claims. However, since applicant believes that the base claims are patentable as discussed above, these dependent claims are also patentable.

IV. CONCLUSION

For all the foregoing reasons, applicant submits that the claims are in condition for allowance, and the examiner's favorable reconsideration to that end is solicited. If additional questions remain, the examiner is encouraged to telephone the undersigned.

Respectfully submitted,

DERGOSITS & NOAH LLP

Dated: 7/28, 2003

By: _____
Richard A. Nebb
Reg. No. 33,540

Four Embarcadero Center, Suite 1450
San Francisco, California 94111
(415) 705-6377 tel
(415) 705-6383 fax
rnebb@dergnoah.com

APPENDIX

1. (currently amended) A software development system for applications that run on a data network which couples a server computer and a client computer, wherein the client computer runs a browser program, comprising

a page generator ~~capable of~~ running an application being developed and sending generated documents to the browser for display as pages ~~generating functional application pages with~~ including additional editing features for interpretation by the browser program;

an editor ~~capable of~~ directly operating on the pages displayed by the browser via the editing features, thereby allowing the user to work on a functional application during development.

2. (currently amended) A software development system as claimed in claim 1, further comprising a plurality of components, and wherein developed applications comprise at least one page template capable of containing components, and wherein the editor provides features to insert, modify and delete components on page templates, and wherein the page generator executes the components on page templates.

3. (currently amended) A software development system as claimed in claim 2, wherein at least one of the components reacts interactively on user input by executing instructions of said component on the server.

4. (original) A software development system as in claim 3, wherein at least one of the components contains at least one other component.

5. (previously amended) A software development system as in claim 3, wherein the set of components on pages generated from a single page template can vary for different requests of the same page template.

6. (currently amended) A software development system for use in a data network which couples a server computer to a client computer, wherein the client computer includes a first software program for generating a request for one or more pages from the

server computer and for displaying pages, and wherein the server computer includes a second software program for receiving and processing the request from the client computer, for generating and storing pages, and for transmitting pages to the client computer in response to requests, the server computer further comprising:

a data store,

a plurality of components residing in the data store, including at least one components that reacts interactively on user input by executing instructions contained in said component on the server;

a plurality of page templates residing in the data store, at least one page template having at least one selected component incorporated therein; and

a server processor controlled by a third software program, said program providing instructions for selecting a page template based on the request from the client computer and instructions for generating a page from the page template for transmission to the client computer.

7. (original) The development system of claim 6, further comprising a component editor controlled by a fourth software program, said program providing instructions for interactively editing selected components on a page template.

8. (original) The development system of claim 6, wherein a component is nested within a component.

9. (previously amended) A method for generating documents for display by a browser using components that react interactively on user input by executing instructions on a server, comprising the following steps for execution on the server upon a document request:

assigning a unique identifier to at least one of the components; and

embedding the unique identifier into a generated document.

10. (previously amended) The method of claim 9, further comprising storing data on the server representing at least one of the components.

APPENDIX

11. (previously amended) The method of claim 10, further comprising:

analyzing the request sent by the browser for unique identifiers; and

calling a function for the interactive components referenced by at least one of the unique identifiers contained in the request.

12. (previously amended) The method of claim 11, wherein at least one of the components is contained on a document template.

13. (original) The method of claim 11, wherein at least one of the components is called by a program.

14. (original) The method of claim 11, wherein at least one of the components is called by a another component.

15. (original) The method of claim 11, wherein the data is stored into an object of an object oriented programming language and wherein the function is a method of the object.

16. (original) A method for implementing client server applications, comprising:

storing data objects on a server and assigning a unique identifier to each data object;

dynamically generating a document with the unique identifier embedded in the document; and

analyzing requests for unique identifiers and calling at least one function for a data object associated with one of the unique identifiers found in the request.

17. (original) The method of claim 16, wherein the unique identifier is embedded inside a uniform resource locator contained in a tag of the document.

18. (original) The method of claim 16, wherein the unique identifier is embedded in scripts contained in the document.

APPENDIX

19. (original) The method of claim 16, wherein the unique identifier is unique within a single session.

20. (previously amended) The method of claim 16, wherein the unique identifier is unique within all documents generated by a single server within a defined time period.

21. (original) The method of claim 16, wherein the data objects are created by an object-oriented programming language and said function is a method of one of these objects.

22. (currently amended) A computer running an application to develop and maintain applications using a web browser, comprising:

an editor operable within the web browser for inserting, deleting, and modifying components on document templates; and

a document generator for processing document templates, executing components and for generating documents from the document templates that are understandable by the web browser.

23. (previously amended) A computer as in claim 22, wherein the editor operates functional applications in an edit mode permitting editing directly in the web browser.

24. (currently amended) A computer as in claim 23 wherein at least one of the components contains instructions and can react on subsequent document requests containing user responses by executing selected instructions of said component.

25. (previously amended) A computer as in claim 24, wherein the computer further comprises:

a store of component classes, each component class implementing one component kind; and

a parser able to detect components marked on document templates;

wherein the document generator works upon a document request using component classes to generate browser code; and

wherein the editor is capable of showing a menu of components for insertion into the document templates.

26. (original) A system to modify documents on a server in a data network which couples said server computer to a client computer, the server computer comprising:

a document store;

a first software program including instructions for transforming a first document retrieved from the document store into a second document having features which permit editing of the first document such that at least a part of the second document appears and functions similar to the first document; and

a second software program including instructions to receive information from the client computer and instructions to modify documents stored in the document store.

27. (currently amended) The system of claim 26, wherein the ~~second~~ first document includes at least one component being executed by the first software program and wherein the second document includes at least one handle to indicate the position of the component to the user.

28. (original) The system of claim 27, wherein the second document includes handles and choosing one of the handles selects an editing operation

29. (original) The system of claim 28, wherein at least one handle indicates the position of at least one component contained in the first document and said editing operation is chosen from the group of modifying the component, deleting the component, displaying information regarding the component, and inserting a new component.

30. (original) The system of claim 26, wherein the features are scripts.

APPENDIX

31. (previously amended) The system of claim 30, wherein the scripts are generated specifically for the second document and encapsulate information which is incorporated into the first document.

32. (original) The system as in claim 26, wherein the features incorporate information regarding the first document into the second document.

33. (previously amended) A system as in claim 32, wherein the information incorporated into the second document is used on the client computer in order to send change requests for the first document to the server.

34. (previously amended) A method for generating a document for display in a browser from a document template containing components, comprising:

for each component denoted on the document template, identifying a component class of the component; and

based on data contained in a request initiated by the browser storing a first object of the component class, the first object representing the component.

35. (previously amended) The method of claim 44, further comprising calling a method of said component class to generate browser code, said method being the constructor.

36. (original) The method of claim 34, further comprising, for all components having a name attribute, looking up the component object in session memory based on said name attribute.

37. (previously amended) The method of claim 34, further comprising, for at least one component kind, for all components denoted on the document template having said component kind;

generating a unique identifier;

assigning said unique identifier to said object, and

embedding said unique identifier into the browser code.

APPENDIX

38. (previously amended) The method of claim 37, further comprising:

inserting objects for the components of at least said component kind into a list of listening components;

working through all objects stored in the list of listening components whose unique identifier occurs inside a name in the form data set; and

calling a method of at least one of these objects.

39. (previously amended) The method of claim 34, wherein the document template is parsed into a list of nodes, including text and component nodes, said method further comprising:

determining if the current node is text or a component;

if component, then calling a method for the component, comprising:

evaluating the attributes of the component if necessary;

identifying the component class associated with the component; and

calling the constructor method of the component class,

said constructor method generating browser code;

if text, then generating the text; and

repeating these steps for each node.

40. (original) The method of claim 39, wherein at least one component contains nested components and the method of claim 39 is recursively performed for all nodes nested inside the component.

41. (previously added) A software development system as in claim 1, the editor comprising a client part for execution on the client computer.

42. (previously added) A software development system as in claim 41, wherein the client part comprises instructions that are automatically downloaded from the server prior to editing.

APPENDIX

43. (previously added) A system as in claim 26, additionally comprising at least one script for automatic download to the client that works in cooperation with the second document to permit editing of the first document.

44. (previously amended) The method of claim 34 wherein storing the first object comprises creating a new object as necessary.

45. (previously amended) The method as in claim 34 wherein components are denoted on document templates using tag syntax.

46. (previously added) The method as in claim 45 wherein the tag name identifies a component class.

47. (previously amended) The method as in claim 36 wherein components are denoted on document templates using tag syntax, wherein the tag name identifies a component class.

48. (previously added) The method as in claim 36 wherein the component object, if found, is reused to store the first object.

49. (previously added) The method as in claim 36 wherein in case a component has a name attribute but no component object is found a new object is created and stored under said name in session memory.

50. (previously added) The method as in claim 49 wherein new objects are created for all components not having a name attribute.

51. (currently amended) A system for editing components on web document templates for use with a first software program including first instructions for generating a document request to obtain at least one generated document from a second software program and for displaying the generated document, the second software program

capable of receiving and processing ~~a~~ the document request and of transmitting first documents to the first software program in response to requests, said system comprising:

a plurality of components encapsulating browser code,

a plurality of document templates,

the second software program transmitting, while processing selected requests, second documents to the first software program that make the first software program display a user interface for editing functions used for maintaining components on document templates,

a third software program used by the second software program while processing selected document requests, the third software program including third instructions for modifying document templates in order to perform said editing functions.

52. (currently amended) The system of claim 51, wherein components include fourth program instructions including steps to generate browser code ~~prior to~~ for transmission to the first software program.

53. (previously added) A system in claim 52 running on a data network, coupling a server computer and a client computer, the first program running on the client computer, the second program running on the server computer.

54. (previously added) A system in claim 52 wherein second documents are HTML pages with embedded scripts.

55. (previously added) The system of claim 52, wherein the edit function includes adding a component to a document template, removing a component from a document template, and modifying attributes of a component on a document template.

56. (previously added) The system of claim 52, further comprising a fifth software program used by the second software program while processing selected document requests, the fifth software program including fifth instructions for generating

generated documents from document templates thereby calling fourth program instructions.

57. (previously added) The system of claim 56, wherein the generated document includes, if requested in edit mode, edit features for interpretation by the first software program.

58. (previously amended) The system of claim 56 further comprising instructions to allow the user to click on the generated document to select items to perform edit functions on.

59. (currently amended) A software development system for developing dynamic web documents for transformation into display documents for display to a user, comprising:

an editor program for editing dynamic web documents,

a document generator for generating generated documents from dynamic web documents which look and function similar to the display documents,

the editor program comprising first instructions for requesting the document generator to process a dynamic web document leading to a generated document,

the editor program further comprising second instructions for displaying at least some information items contained on said generated document in a view which allows the user to select an item to which a modification function will be applied,

the editor program further comprising third instructions to modify the dynamic web document to perform said modification function.

60. (previously added) The software development system of Claim 59 running on a data network, which couples a server computer and a client computer, the document generator running on the server computer the editor at least partly running on the client computer.

APPENDIX

61. (previously added) The software development system of claim 60, wherein the document generator further comprises fourth instructions for execution during document generation to collect edit-information for use by the editor.

62. (previously added) The software development system of claim 60, wherein the editor uses a web browser for displaying said view.

63. (previously added) The software development system of claim 60, able to automatically repeat requesting the document generator to process the dynamic web document if required.

64. (previously added) The software development system of Claim 59 further comprising a plurality of components marked on the dynamic web document, components including instructions for use by the document generator to generate browser code.

65. (previously added) The software development system of claim 64, wherein the editor uses a web browser for displaying said view.

66. (previously added) The software development system of claim 64, wherein modification functions include insert of a component, delete of a component, and modify attributes of a component.

67. (previously added) The software development system of claim 59, wherein said view looks, except for editing features, similar to the end-user view of the generated document.

68. (previously added) The software development system of claim 59, wherein the document generator further comprises sixth instructions to collect edit-information for use by the editor, said sixth instructions for execution during document generation.

APPENDIX

69. (previously added) The software development system of claim 68, wherein the editor uses the edit-information to correctly modify the dynamic web document.

70. (previously added) The software development system of claim 69, further comprising a plurality of components marked on the dynamic web document, wherein the edit-information comprises position information on the components contained in the document template.

71. (previously added) The software development system of claim 59, wherein the editor uses a web browser for displaying said view.

72. (previously added) The software development system of claim 71, wherein first instructions comprise seventh instructions for initiating a reload in the browser.

73. (previously added) The software development system of claim 59 the editor program further comprising eighth instructions to display information on at least one element of the dynamic web document, that is replaced during document generation, without requesting the document generator to regenerate the generated document.

74. (currently amended) A software development system for document templates that are intended for transformation into generated documents for display by a first software program, the first software program including first instructions for generating a document request to obtain at least one generated document and for displaying the generated document, comprising:

a plurality of components, that include instructions to generate browser code ~~prior to~~ for transmission to the first software program,

an editor capable of performing edit functions maintaining components on document templates, the components capable to cooperate with the editor,

a plurality of document templates having components denoted thereon, and

a document generator comprising second instructions to, upon a document request, generate generated documents from document templates for display by the first

software program wherein the set of components on the generated document can vary for different document requests for the same document template.

75. (previously added) The software development system as in claim 74, wherein edit function comprises adding a component, modifying a component, and deleting a component.

76. (previously added) The software development system as in claim 74, wherein tag syntax is used to denote components on document templates, whereby the tag name identifies the component kind.

77. (previously added) The software development system of claim 74 running on a data network, which couples a server computer and a client computer, the document generator running on the server computer the editor running, at least partly, on the client computer.

78. (previously added) The software development system as in claim 74, wherein a component, that can react interactively on subsequent document requests, can be excluded from the generated document.

79. (previously added) The software development system as in claim 78 further comprising third instructions to prevent excluded components from reacting on subsequent document requests.

80. (previously added) A software development system as in claim 79, third instructions comprising fourth instructions to, upon a first document request, store information in session memory on all components, that are present on the generated document, and fifth instructions to, upon subsequent document requests, only react on components that have been remembered in session memory thereby avoiding tampering with excluded components on the side of the first program.

APPENDIX

81. (previously added) A software development system as in claim 74 wherein at least one document template has a first and a second component denoted thereon in a way that the first component contains the second component, the first component containing sixth instructions to decide about exclusion of the second component from the generated document.

82. (previously added) A software development system as in claim 74 the editor able to provide an editable view taking the varying set of components into account.

83. (previously added) A software development system as in claim 74 the editor able to provide an editable view that includes and excludes selected components similarly as the final application.

84. (previously added) A software development system as in claim 74 wherein the generated document contains more components than the document template for at least one document request.

85. (previously added) The software development system as in claim 74, wherein multiple instances of a third component denoted on the document template can be included in the generated document.

86. (previously added) The software development system as in claim 85, further comprising seventh instructions to assign a unique identifier to each component instance, whereby the third component includes eighth instructions to qualify names generated into the browser code with the unique identifier.

87. (previously added) A software development system as in claim 74, wherein at least one document template has a fourth and a fifth component denoted thereon in a way that the fourth component contains the fifth component, the fourth component containing ninth instructions to decide about how many instances of the fifth component are included into the generated document.

APPENDIX

88. (previously added) A software development system as in claim 74 the editor able to provide an editable view that include multiple instances of components similarly as the final application.

89. (previously amended) A software development system as in claim 74 wherein at least one sixth component includes tenth instructions to display the sixth component, the tenth instructions being used to generate browser code for displaying the sixth component during editing as well as during normal use of the component.

90. (currently amended) An editor for use with a web browser, the editor allowing the user to edit a document displayed by the browser, wherein clicking on said document displayed in the browser initiates editing functions, and scripts contained in said document remain functional during editing, the editor including a first software program for execution within the browser and for processing the clicking on said document.

91. (previously added) The editor as in claim 90 using at least two windows, a first browser window displaying said document and a second window for displaying information on an element contained in said document.

92. (previously added) The editor in claim 90 further comprising a second software program for modifying documents in cooperation with the first software program.

93. (previously added) The editor as in claim 92 further comprising a third program for transforming the document into a generated document thereby adding editing features, the browser displaying the generated document looking similar to the original and interpreting the editing features.

APPENDIX

94. (currently amended) The editor as in claim 93 wherein said document is a dynamic document having components denoted thereon, the third software program further comprising instructions for generating browser code in cooperation with ~~for~~ components.

95. (previously added) The editor as in claim 94 wherein the browser together with the first software program is running on a client computer connected to a server computer via a data network, wherein the second and the third software program run on the server computer.

96. (previously added) The editor in claim 90 wherein links contained in said document stay functional allowing the user to browse and edit at the same time.

97. (previously amended) A system for displaying dynamically generated documents in a data network coupling a server computer to a client computer, wherein the client computer has a first software program including first instructions for generating a document request to obtain at least one generated document from the server computer and for displaying the generated document, comprising:

a plurality of components for execution on the server computer, including a first component including second program instructions to generate browser code and third program instructions for execution on the server which are initiated by the user interacting with the first component, and,

fourth program instructions on the server computer for, based on data contained in a document request initiated by the first software program on the client computer, generating generated documents for transfer to the client computer and display by the first software program, thereby calling second program instructions of components.

98. (previously added) The system of claim 97, the server computer further comprising fifth program instructions for analyzing said data and for calling third program instructions of first component as necessary.

APPENDIX

99. (previously added) The system of claim 98, further comprising a plurality of document templates residing in the data store, at least one of the document templates having at least one second component denoted thereon.

100. (previously added) The system of claim 99, wherein tag syntax is used to denote the second component, whereby the tag name identifies a component.

101. (previously added) The system of claim 99, wherein at least one third component denoted on a document template contains the first component.

102. (previously added) The system of claim 101, wherein third components implementation scheme includes logic to decide how often to insert the first component into the generated document.

103. (previously added) The system of claim 98, the server computer further comprising sixth program instructions for, based on the document request, deciding to insert more than one instance of the first component into the generated document.

104. (previously added) The system of claim 103, making sure that multiple instances of the first component do not interfere by qualifying names generated into the browser code using unique identifiers.

105. (previously added) The system of claim 98, the server computer further comprising seventh program instructions for, based on the data, deciding to exclude the first component from the generated document.

106. (previously amended) The system of claim 105, wherein fifth instructions call third instructions only if the first component was contained on a page previously transferred to the client.

APPENDIX

107. (previously added) The system of claim 98, wherein fifth program instructions include eighth program instructions to analyze said data for user interactions with multiple components and to call third program instructions of multiple components as necessary.

108. (previously added) The system of claim 107, wherein components include ninth instructions to check for errors and fifth instructions include tenth instructions to call ninth instructions of components as necessary and to suppress subsequent calling of third instructions in case of errors.

109. (previously added) The system of claim 98, further comprising eleventh program instructions for storing a data object in session memory representing at least one component instance included in a generated document, said eleventh program instructions for execution while dynamically generating a document.

110. (previously added) The system of claim 109, wherein third program instructions are encapsulated in a method of data objects, fifth program instructions including twelfth program instructions for identifying the data object that represents a component instance the user interacted with and for calling said method of said data object as necessary.

111. (previously added) The system of claim 110, further including thirteenth instructions for deciding based on said data to include more than one instance of a component into the generated document.

112. (previously added) The system of claim 109, further comprising twelfth program instructions for assigning a unique identifier to the first component instance, for associating the unique identifier with the data object, and for including the unique identifier into the generated document, said twelfth program instructions for execution while dynamically generating a document .

APPENDIX

113. (previously added) The system of claim 112, wherein fifth program instructions analyze said data for unique identifiers and include thirteenth instructions for identifying the associated data object.

114. (previously added) A system for displaying dynamically generated documents in a data network coupling a server computer to a client computer, wherein the client computer has a first software program including first program instructions for generating a request to obtain at least one generated document from the server computer and for displaying the generated document, comprising:

a plurality of components for execution on the server, at least one of the components including first features to cooperate with an editor in editing said component and second program instructions to generate browser code, and

third program instructions on the server for, based on the data contained in a request initiated by the client computer, generating generated documents for transfer to the client computer, thereby calling second program instructions of components.

115. (previously added) The system of claim 114 wherein first features include fourth program instructions for passing information to the editor.

116. (previously added) The system of claim 115 wherein at least part of said information is collected during execution of the components on the server.

117. (previously added) The system of claim 115 wherein said information is transmitted from the server to the client.

118. (previously added) The system of claim 115 wherein said information includes attributes of said component.

119. (previously added) The system of claim 114 wherein first features include fifth instructions that display additional editing features of the component during editing.

APPENDIX

120. (previously added) The system of claim 119 wherein said editing features include handles.

121. (previously added) The system of claim 114 wherein first features include an extension for use by the editor, said extension for enabling editing of the components attributes.

122. (previously added) The system of claim 121 wherein said extension is a page for editing components attributes.

123. (previously added) The system of claim 114 wherein components are denoted on document templates using tag syntax, whereby the tag name identifies a component.

124. (previously amended) The system of claim 114 containing at least one component wherein second program instructions are used to generate browser code for displaying the component during editing and during normal use.

125. (currently amended) A method for editing an application that is built using components and that operates by generating documents comprising the steps of:

running the application, thereby executing components and generating a generated document,

displaying a view of the generated document,

selecting a component by clicking on selected portions of said view,

identifying the selected component in the source code of the application,

initiating a modification function modifying the source code of the application.

126. (previously added) The method of claim 125 wherein the running step and the displaying step are repeated after applying a modification function.

APPENDIX

127. (previously added) The method of claim 125 further comprising collecting edit information for use by the identifying step.